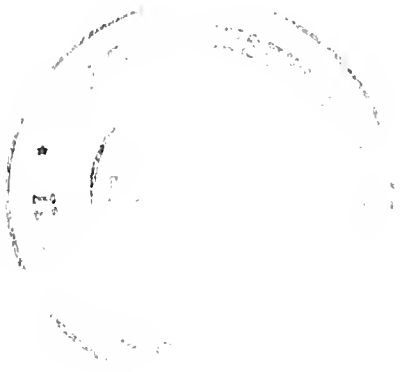


A Case-based System to Support Electronic Circuit Diagnosis

Thilo Semmelbauer,
Anantaram Balakrishnan,
and
Haruhiko Asada

WP# 3497-92-MSA October, 1992

A Case-based System to Support Electronic Circuit Diagnosis

Thilo Semmelbauer,
Anantaram Balakrishnan,
and
Haruhiko Asada

WP# 3497-92-MSA October, 1992

A Case-based System to Support Electronic Circuit Diagnosis †

Thilo Semmelbauer
Leaders for Manufacturing
M. I. T., Cambridge, MA

Anantaram Balakrishnan
Sloan School of Management
M. I. T., Cambridge, MA

Haruhiko Asada
Department of Mechanical Engineering
M. I. T., Cambridge, MA

October 1992

† Supported by MIT's *Leaders for Manufacturing Program*

Abstract

Diagnosis and repair operations have traditionally been major bottlenecks in electronics circuit assembly operations because they are labor intensive and highly variable. Rapid technological advances, shrinking product lifecycles, and increasing competition in the electronics industry have made quick and efficient diagnosis critical for cost control and quality improvement, while simultaneously increasing the difficulty of the diagnosis task. This paper describes a case-based diagnosis support system to improve the effectiveness and efficiency of circuit diagnosis by automatically updating the set of candidate defects, and prioritizing tests during the sequential testing process. The case-based system stores individual diagnostic instances rather than general rules and algorithmic procedures; its knowledge base grows as new faults are detected and diagnosed by the analyzers. The system provides distributed access to multiple users, and incorporates on-line updating features that make it quick to adapt to changing circumstances. Because it is easy to install, update, and integrate with existing systems, this method is well-suited for real manufacturing applications. We have implemented a prototype version, and tested the approach in an actual electronics assembly environment. We describe the system's underlying principles, discuss methods to improve diagnostic effectiveness through principled test selection and sequencing, and discuss managerial implications for successful implementation.

1. Introduction

1.1 Overview

Electronic circuit diagnosis is the process of identifying the component(s) that is responsible for the malfunction of a circuit board so that corrective (repair) action can be taken. Subsequent exploration of the problem's root causes motivates process and product improvement efforts. The diagnosis and repair loop has traditionally been a major bottleneck in circuit board assembly. Recent trends in the electronics industry such as the rapid technological advances, increasing product complexity, and short product lifecycles, have considerably increased the importance of quick and efficient diagnosis, while simultaneously increasing the difficulty of the diagnosis task. Reducing diagnosis time and variability, and improving diagnosis accuracy becomes critical as electronics companies strive towards lean manufacturing, quick response, and greater flexibility.

This paper describes a case-based diagnosis support system that we developed in consultation with a high-volume consumer electronics assembly plant. The plant, faced with a rapid turnover of products and an increasing shortage of skilled technicians, wanted a means to improve the efficiency and effectiveness of circuit diagnosis, reduce the workload on its few expert analyzers, and develop a tool to train new technicians, without devoting significant system development resources. An informal survey of available systems and current practice revealed that, although the literature describes several rule-based and model-based systems to support diagnosis, these systems were not widely used by the electronics industry partly because they require considerable time and special expertise to install (e.g., for knowledge acquisition) and update. The case-based approach proposed in this paper offers a practical solution to the diagnosis support needs of industry. It accumulates knowledge on-line as analyzers detect and diagnose new faults. Using this information about prior diagnostic instances, the system eliminates candidate defects during the sequential testing process, and prioritizes subsequent tests. We have implemented a prototype version of the system, and tested it in an actual electronics assembly environment. We describe the system's underlying principles, outline methods to improve diagnostic effectiveness by reducing the number ot tests needed to complete the diagnosis, and discuss implementation aspects.

1.2 Why is circuit diagnosis important?

Rapid changes in electronics product and process technology, such as smaller, surface-mounted components, higher board densities, and more complex circuitry, have made circuit diagnosis more challenging. With fewer accessible connections and test pads, in-line functional testing can only identify the approximate area or block of components on the

board that is responsible for the malfunction. Detailed diagnosis must be done off-line through extensive probing and measurement.

Each component and connection on a circuit board is a potential defect site, and every site can have several possible defects such as an open or short circuit (due to poor soldering or component misalignment), component malfunction, inherent flaw in the circuit design, placement of the wrong component, or defect in the board's internal wiring. The failure rate of a board containing hundreds of components is, therefore, several orders of magnitude higher than the probability of a single defective component or operation, making testing and diagnosis inevitable even with highly capable placement and soldering operations. Isolating the root cause in such complex and dense circuits is arduous and time consuming. Annual expenses directly related to diagnosis and repair can exceed several hundreds of thousands of dollars even in a medium-sized facility. Work in process at the diagnosis stage and delayed feedback for process control further increase this cost. More importantly, the time and effort required to diagnose a faulty board is highly variable (from a few minutes to over 4 hours) depending on the type of defect, the analyzer's skill, the effectiveness of diagnostic tools, and the incentive and performance evaluation systems. Consequently, testing, fault diagnosis and repair operations are often major bottlenecks in medium or high volume printed circuit board assembly facilities.

The cost and effort to train technicians is also becoming a significant concern for electronics companies. A new technician requires several months of training and experience with a particular board to understand the circuit's cause-effect behavior and the potential root causes in its manufacturing process. As product life cycles shrink, the training costs must be amortized over smaller volumes for each board. Diagnosis bottlenecks are especially acute during the critical production ramp-up period following a new product introduction. Capturing market share early is essential for competitive survival in the electronics industry, but production facilities often cannot achieve the necessary rapid ramp-up because yields are low, and analyzers are least productive since they have not had significant experience with the new product.

Fast and accurate diagnosis not only reduces direct manufacturing cost and flow time, but is also an essential prerequisite for successful quality improvement efforts through improved product design (e.g., Soederberg [1992]), better process understanding and characterization, tighter process control, and close supplier partnership (e.g., Wenstrup [1991]). The ability to provide quick information feedback from the diagnosis stage to previous stages (including vendors) is particularly important in medium and high volume environments. For instance, a stuck glue pin or a worn placement head, if not detected early, can produce hundreds of defective boards.

1.3 Computer support for circuit diagnosis

Manual diagnosis of circuit boards can be both inefficient and error-prone. Empirical evidence (Rouse and Morris [1985], Rouse and Hunt [1986]) suggests that humans are not optimal diagnosticians since they have difficulty in effectively utilizing all the available test data to prune the list of candidate defects and systematically select the test sequence. On the other hand, building a completely automated system that is capable of anticipating and diagnosing all possible defects can be prohibitively expensive; such a system might also require extensive time and effort by specialists (design and process engineers or software experts) to install and maintain. Therefore, a system that assists, rather than replaces, human diagnosticians is more practicable. To cope with the dynamic electronics environment, the system must be robust and flexible, i.e., it should easily adapt to design and process changes. Practitioners prefer a flexible system that possibly diagnoses only the most common defect types (that account for a vast majority of board failures) to one that offers comprehensive diagnostic coverage but cannot easily adapt to changes.

The case-based approach offers the potential to meet these requirements. Unlike other knowledge-based systems, the case-based system accumulates knowledge on-line by storing diagnostic instances (as analyzers identify and diagnose new faults) rather than general rules or algorithmic procedures. Each diagnostic instance or case corresponds to a distinct defect; it is represented by the set of tests conducted and their observed outcomes for that defect. By comparing the test results for the current board with previous instances, the system can eliminate candidate defects and determine the effectiveness of subsequent tests. Recording diagnostic instances rather than general rules has the potential disadvantage of requiring a large database. However, our prototype implementation experience suggests that the vast majority of faults encountered in practice correspond to a very small fraction of the set of all possible instances.

Because of its simplicity, the system can be maintained by the analyzers themselves, avoiding long knowledge-acquisition delays and the consequent system obsolescence. Furthermore, the standard format of a case promotes learning and information sharing among technicians (from different shifts or production lines), and enables the system to serve also as a training tool for new technicians. These features make the case-based system viable and useful for real manufacturing applications. Although case-based reasoning has been applied to other domains such as legal argument and customer service, we are not aware of its previous application to electronic circuit diagnosis. We enhance the basic case-based approach to incorporate historical data and other prior information for systematically selecting tests and optimizing the search process. To validate the overall approach, we implemented and tested a prototype version of the system in a high-volume printed circuit board (surface-mount) assembly line producing a hybrid (analog+digital) circuit board for a state-of-the-art consumer electronics product. Our implementation

permits distributed usage via a local access network. Within five weeks of installation, the system achieved 90% coverage, i.e., the system had acquired enough diagnostic instances to correctly diagnose 90% of the defects occuring in the fifth week. Based on our implementation experience, we offer suggestions to improve the management of the diagnosis function; in particular, we propose a proactive management approach and incentive schemes that emphasize the analyzer's process improvement rather than fault-finding role.

The rest of this paper is organized as follows. Section 2 describes the circuit diagnosis process in more detail, and develops the goals for an ideal diagnosis support system. Section 3 discusses the underlying structure and operation of a basic case-based system for verifying defects based on test outcomes. Section 4 presents system enhancements to optimize test selection using historical data and other prior information on the likelihood of different defects. We present a dynamic programming formulation of the optimal test sequencing problem to minimize the expected number of tests per board, and outline heuristic methods for on-line test selection. Section 5 briefly describes our prototype implementation and the lessons we learned from this implementation exercise, and identifies directions for further work.

2. The Diagnosis Process

2.1 In-line testing: Capabilities and tradeoffs

Circuit board assembly lines typically contain in-line inspection and testing stages, after placement and soldering operations (see, for instance, Noble [1989] for a description of the processing steps in printed circuit board assembly). The main purpose of these in-line screening operations is to identify defective boards, although they also have limited diagnostic capabilities. For instance, visual and X-ray inspection can identify missing or misplaced components and possibly certain solder defects, but are not effective or reliable for dense boards with small components. Furthermore, complete manual or X-ray inspection can be very time consuming and expensive.

Two types of in-line electrical tests are available—in-circuit testing and functional testing. In-circuit tests verify the functionality of individual components on the board using automated test equipment (see, for instance, Maunder and Tulloss [1990]). They can isolate and precisely identify certain types of defects (e.g., open or short circuits, defective passive components), but require hard fixturing and programming that is difficult to justify for short production runs. Moreover, the increasing use of surface-mount technology and the high cost of board "real estate" make in-circuit testing infeasible or limit their diagnostic capabilities (Garcia-Rubio [1988]). In-circuit tests are also prone to measurement errors (see, for example, Chevalier [1992]) especially for small, high performance components,

leading to false acceptance or rejection of boards. Functional tests measure the circuit's response (at the board's output ports or edge connectors) to different combinations of input signals. Functional tests can confirm whether the interactions between various components meet the specifications; typically, they can trace a board's failure to the responsible circuit, but not to a particular component or connection in this circuit.

Selecting the testing strategy for a product involves addressing tradeoffs between higher assembly cycle times (for all boards) to conduct extensive in-line diagnostic tests versus higher off-line diagnosis effort (for defective boards) when the in-line tests do not have high resolution. As the board defect rate decreases and component density increases, this tradeoff increasingly favors the strategy of using in-line testing only to identify defective boards; detailed diagnosis and defect verification is then performed off-line by skilled technicians. We next describe this off-line diagnosis process, and identify opportunities to support this process using a computer-based system. For brevity, we will use the word "diagnosis" to refer to the detailed, off-line diagnosis procedure after a board has failed in-line (functional) tests.

2.2 Sequential testing procedure for off-line diagnosis

Diagnosing a circuit board that has failed functional tests entails identifying the source of the malfunction (e.g., a component or connection on the board) so that appropriate corrective action (e.g., replacing or resoldering the component) can be taken to repair the board. Diagnostic information also serves as the input for root cause analysis and continuous improvement of vendors, processes, and product design.

We define a **defect** as the finest grain source of malfunction that can be unambiguously identified by the analyzer. Each defect has an associated **corrective action**; different defects might require the same corrective action. For instance, a defective resistor can act as either a short circuit or an open circuit. In each case, the symptoms and the means to identify the problem are different; therefore, we view these two possibilities as separate defects even though both require the same corrective action (namely, replacing the resisitor). Each defect has a set of associated **symptoms** that the analyzer must discover by applying one or more **tests**. A test is an action that must be performed either by a technician or by automated test equipment, resulting in an observation about the behavior of the board. A test might consist of applying a specified set of electrical inputs, adjusting certain circuit elements (e.g., tuning capacitors), and probing selected locations on the circuit board to measure the voltage or view the electrical signal on an oscilloscope. Each test has two or more possible results or **outcomes** (e.g., voltage is greater than 4.5 volts or less than 4.5 volts); observe that if the output measurement is continuous (e.g., voltage) we discretize it by defining the range of continuous values associated with each outcome. For convenience, and without loss of generality, all of our subsequent discussions assume

that every test has two possible outcomes which we denote as 0 or 1. The outcome of a test depends on the defect in the board. We permit defects to have indeterminate outcomes for certain tests, e.g., for a certain defect, the voltage at a particular location may be either 0 or 1 depending on, say, the state of a flip-flop device. We also recognize that the analyzer might have only partial prior information regarding which defect produces what outcome for each test; however, we assume that the available information is adequate to uniquely identify each defect.

To identify the true defect in a malfunctioning board, the analyzer uses a **sequential testing procedure**. The process starts with a candidate set of defects (which depends on the board's failure mode during functional testing) and a set of tests that can distinguish between these defects. The analyzer successively selects and applies various tests, observes the outcome at each stage, and eliminates (from the candidate defect set) all defects that cannot produce the observed outcomes. If the initial candidate defect set is complete (i.e., the set contains all possible defects), and the set of available tests is comprehensive (i.e., some combination of known test outcomes uniquely identifies each candidate defect), the procedure terminates with the true defect as the only remaining member of the candidate defect set.

In practice, analyzers might not always follow a systematic procedure to select tests and eliminate defects. Diagnosis is often considered an "art", and analyzers rely largely on intuition, some circuit knowledge, memory, and experience as they perform diagnosis. Often, they do not formally maintain and update the set of candidate defects by comparing the defects' expected test outcomes with the observed outcomes. Indeed, the set of candidate defects and their expected outcomes may not even be documented, and might vary from technician to technician. A common "myopic" approach for troubleshooting consists of hypothesizing a particular defect, attempting to verify it by performing the appropriate tests, revising the guess if the tests disprove the first hypothesis, and so on. Alternatively, the analyzer might follow a predetermined test sequence represented, for instance, as a fault tree.

A purely manual diagnosis process can be both inefficient and inaccurate especially when it is not supported by clear procedures and documentation. For instance, using a static fault tree can result in unnecessary tests and unproductive search (relative to an approach that uses recent defect history to guide the search). Likewise, diagnosis without adequate documentation and formal recording procedures can lead to replication of tests and false identification. Also, these informal approaches are not conducive to learning, transfering knowledge to other analyzers, or generating appropriate information for quality improvement. To identify the desirable characteristics of a computer-based system to assist analyzers, let us first explore the available opportunities to improve diagnosis productivity.

2.3 Diagnosis decision support requirements

Consider an intermediate stage in the diagnosis process when the analyzer has applied several tests and observed the results of each test. The analyzer now faces four related questions, all aimed at deciding what to do next:

(1) Given the outcomes of all previous tests, can we conclude that the board contains a specific (known) defect?

(2) Does the board contain a previously unknown defect?

(3) If more than one (known) defect remain as candidates, which available test(s) can distinguish between these defects?

(4) What test to apply next?

We will refer to the first two questions as **Defect Identification**, and the remaining two as **Test Selection**.

Consider the information needs and actions for the defect identification questions. An affirmative answer to either question takes the board out of the regular diagnosis iteration. In the first case, the action consists of repairing the identified defect, possibly after confirming the diagnosis. If no known defect causes the observed cumulative test outcomes (i.e., the answer to question 2 is affirmative), we have three possibilities: (i) the board contains a new defect, or (ii) the assumed test outcomes for the known defects are inaccurate, or (iii) the actual test outcomes were measured incorrectly. In all three situations, the board requires extraordinary actions such as confirming previous test outcomes, consulting with design and process engineers, checking with suppliers, and updating current knowledge.

If multiple (known) defects remain in the candidate set, the test selection questions first seek to identify relevant tests, not yet performed, that can distinguish between the remaining possible defects. Conceptually, this process corresponds to determining, for each available test, if that test has different expected outcomes for the candidate defects. If no such test is available, the analyzer must develop and apply one or more new tests to distinguish the defects. Otherwise, we must decide which among the available tests to perform next. As we shall see later, this decision impacts the number of iterations needed to complete the diagnosis.

Computer support for the defect identification questions might consist of a database or logical reasoning system that automatically compares the observed test outcomes with the expected outcomes for all candidate defects, and eliminates defects with inconsistent outcomes. Similarly, to support test selection, a computer-based system can identify the relevant tests at each stage, and apply a systematic method to select the next test. To be effective, the diagnosis support system must meet certain key requirements such as ease of

initializing and updating, and quick response. We list below these and other desirable features of a computer-based system to support electronic circuit diagnosis:

- *Development effort*: Given the short product lifecycles of electronic products, we require a system that becomes operational quickly and without significant expense.
- *Maintainability/robustness:* The system must be easy to update, preferably by the users themselves; relying on external resources to update the system (e.g., software experts and knowledge engineers) often introduces delays, causing inconsistencies and system obsolescence. Another important requirement is the system's ability to adapt gracefully to the frequent design and process changes that characterize the electronics industry.
- *Compatibility*: Technicians often view structured tools and methodologies as impediments to creativity; compatibility with current practices and the culture of the organization is, therefore, critical to the success and effectiveness of the system. System ergonomics, ease of use, flexibility, and manual override options are some of the ways to improve compatibility.
- *Response time/efficiency*: Since each board requires several test iterations to prune the candidate defect set, the system must respond rapidly to these questions so that diagnosis does not become the bottleneck activity.
- *Effectiveness*: The main purpose of a diagnosis support system is to reduce the diagnosis time per board. Part of this savings comes from eliminating errors and automating some of the routine bookkeeping and lookup functions. Two other features can reduce diagnosis time:
  - reducing the (expected) number of tests through systematic search and principled test selection and sequencing; and,
  - providing good "coverage", i.e., when the system has stabilized, it must correctly diagnose a large percentage of the defects.

Other desirable system features include abilities to: interface with existing databases, prepare periodic summary reports for process improvement, highlight inconsistencies and facilitate learning, diagnose both digital and analog (e.g., radio frequency) circuits, and operate in multi-user environments. The next section uses these performance criteria to motivate the case-based method.

3. The Case-based Approach for Circuit Diagnosis

3.1 Knowledge base for diagnostic systems

Circuit diagnosis requires three broad classes of knowledge—historic, heuristic, and fundamental knowledge. *Historic* and other prior information on the likelihood of defects can help the analyzer prioritize candidate defects, thus enabling an effective choice of test sequence. For example, if a placement machine frequently misplaces a particular part, the

analyzer might first examine that part before performing other tests. *Heuristic* or *empirical* knowledge, sometimes termed "shallow" knowledge, refers to an empirical association between observed symptoms and diagnostic conclusions. This category might include rules and procedures, as well as tricks or shortcuts that the analyzer has found to be effective after some troubleshooting experience. *Fundamental* or "deep" knowledge implies an understanding of the underlying physics of circuit elements which we use to predict circuit response. Design and test engineers rely on such fundamental knowledge, often encoded in a circuit simulator, to design and troubleshoot electronic circuits.

Depending on the primary source of diagnostic knowledge, we can distiguish between experience-based (empirical knowledge) and model-based (fundamental knowledge) systems to support circuit diagnosis. The *model-based* approach uses an analytical model or a circuit simulator to dynamically identify the potential causes for the observed symptoms during the sequential testing process. Since diagnosis entails "backtracking" (i.e., given certain observed symptoms or outputs, what values of circuit elements can produce these outputs), we need a translator to convert traditional circuit design models (that characterize the output for a given circuit configuration) into corresponding diagnostic models. Davis and Hamscher [1988] illustrate the three basic steps—hypothesis generation, hypothesis testing, and hypothesis discrimination—in model-based diagnosis. Gensereth [1984], deKleer and Williams [1987], and Hamscher [1988] describe various model-based systems to diagnose digital circuits. The most common *experience-based* approach is a rule-based (expert) system that relies on the experience of human experts to develop if-then rules that enable the system to reason about the behavior of the malfunctioning board (see Semmelbauer [1992] for an illustration of this approach). One of the pioneering applications of the rule-based approach was for medical diagnosis (e.g., see Harmon and King [1985]). Unlike the field of medicine, however, electronic circuits and manufacturing processes vary widely, and the technologies undergo frequent and radical changes. In general, the literature on knowledge-based diagnostic systems focusses on knowledge representation issues and inference mechanisms to identify the underlying source of the problem, and does not adequately address issues of optimizing the sequential search process.

For electronic circuit diagnosis, both the model-based and rule-based approaches offer certain advantages, but also impose limitations. By considering a decision support framework that combines both approaches in a complemetary fashion, we can exploit their respective strengths and overcome the disadvantages. Model-based systems offer the advantage of directly integrating design and diagnosis: when the designer changes the circuit schematic (and adds relevant sub-models for new components), the diagnosis model is automatically updated, permitting instantaneous troubleshooting capability without requiring any human experience with the new circuit (Davis [1988]). However, because

they are very computationally intensive, model-based diagnostic systems are not well-suited for on-line applications. Furthermore, technical difficulties still remain in building suitable diagnostic models for analog circuits (see, for example, Bennetts [1981], Kritz and Sugaya [1986], Tong et al. [1989]) due to bi-directional signal propagation, and the impact of physical layout and tolerances on the performance of high speed radio-frequency circuits. Dealing with bridge defects and improper (open) connections also poses problems since the basic circuit structure itself changes (Priest and Welham [1990]). On the other hand, the rule-based approach can incorporate heuristic knowledge to deal with these complexities, and provide quick on-line response during diagnosis; the infrastructure of tools to implement a rule-based system is also quite well-established. However, traditional expert systems might require long development times (Priest and Welham [1990]), especially for initial knowledge acquisition (e.g., Newstead and Pettipher [1986]). Furthermore, since each component and connection on the board can cause malfunction, anticipating and encoding rules for every possible defect is almost impossible. For instance, the initial knowledge that we acquired for one board to build a prototype system covered less than 20% of all the defect types that were encountered during the first five weeks of system operation. Finally, a system that relies on software experts or knowledge engineers to update the knowledge base (for instance, when the circuit design or manufacturing process changes) can introduce expensive delays.

To cope with the rapid pace of change in the electronics industry, we require a diagnostic system that can both adapt easily to the changing environment, and reduce diagnosis time in production by providing quick on-line response. Next, we describe a case-based approach that is similar to rule-based systems in its on-line operation, but can exploit the capabilities of model-based systems to adapt to changing circuit designs and new defects.

3.2 Elements of case-based reasoning

The goal of our project was to develop and test a diagnostic support system that is viable in a practical manufacturing environment. Two important observations regarding practice motivated our proposed case-based approach:

1. The ubiquitous 80-20 law applies also to electronic circuit diagnosis, i.e., less than 20% of all possible defects occur in over 80% of the defective boards. Figure 1, which shows the cumulative relative frequencies of various defects (arranged in decreasing order of occurence) for our prototype system, clearly illustrates this phenomenon. Given this 80-20 characteristic, a system that can operate with incomplete knowledge, progressively adding defects as they are encountered, is preferable to enumerating, a priori, all possible defects. This strategy has four important advantages: (a) the developmental effort and time is likely to be significantly lower; (b) focusing on the relatively few defects that have

occured in the past can significantly reduce the response time; (c) since it can tolerate incomplete knowledge and can be updated on-line, the system can easily adapt to changes in process and product design; and (d) by relying on human experts to diagnose new defects, the system is non-threatening and hence likely to be well-accepted. Our experience (reported in Section 5) indicates that the strategy of progressively adding new defects as they are encountered provides over 90% coverage within just a few weeks of operation.

2. A simple way to "learn" from an expert analyzer is by recording the actions—the tests performed and their respective outcomes—as the analyzer explores and identifies a new defect. The observed outcomes for all the tests that the analyzer applied to diagnose the board characterize the new defect, making it a convenient knowledge representation for system maintainability, and as a means to communicate among experts.

Case-based reasoning (CBR) originated in the area of legal argument (hence, the term "case"). An early system, HYPO, operated in the common law domain of trade-secret law; the system combines reasoning about the statutes (rules) with the precedents (cases) that concern them. The problem of finding the case(s) appropriate to the current situation amounts to clever searching of the case base on a number of significant attributes. The "closest" cases with respect to these attributes are selected for perusal by the researcher. Legal argument continues to dominate the most recent case-based reasoning literature (see, for example, Rissland and Skalak [1988]). Applications of CBR to the fault diagnosis are limited. Lee and Liu [1988] apply CBR to robotic assembly cell diagnosis, and Ishida and Tokumaru [1990] present a generalized approach to case-based diagnosis using the frame theory. For electronic circuit diagnosis, the case-based approach offers the attractive features of compatibility and incremental, on-line knowledge acquisition; the system can tolerate incomplete knowledge, and its coverage increases with time. We enhance the conventional case-based approach by adding features to incorporate:

(i) *robust knowledge acquisition*: the system identifies incomplete and inconsistent knowledge (e.g., due to measurement errors or design changes), and incorporates verification and updating steps to correct these errors;

(ii) *distributed usage*: by implementing the system in a distributed client-server environment with a shared database, several users can simultaneously access and use the system; and,

(iii) *effective test sequencing*: the system reduces the number of tests by applying test selection heuristics that account for prior probabilities of different defects.

In the electronic circuit diagnosis context, a **case** is an instance of a diagnostic experience, represented by its test outcomes. (A rule, by contrast, is a generalization about diagnostic experience gathered over time.) Each case corresponds to a particular defect.

The **case base** consists of the collection of instances corresponding to different defects observed in the past. Recall that each defect has either a deterministic or indeterminate outcome for every test; further, the value of a deterministic outcome might be known (0 or 1) or unknown. For a given defect, we refer to its set of expected outcomes for all tests as the **signature** of that defect. Suppose we have **n defect types** and **m available tests**. We visualize the case base as a n x m matrix, with each row corresponding to a previously diagnosed case or defect. The i^{th} row in the case base represents the **signature** of the i^{th} defect type, i.e., the j^{th} element of this row has a 0, 1, I, or U depending on whether the expected outcome of test j for defect i is 0, 1, indeterminate, or unknown. We permit the user to dynamically add rows (cases) and columns (tests) to the case base as new defects are detected and diagnosed (incremental knowledge acquisition).

3.3 Diagnosing a defective board: Successive refinement through case matching

Diagnosing a defective board corresponds to gathering enough information (by performing tests and observing their outcomes) about the board's signature, while simultaneously searching the case base, to identify a matching case. If no matching case is found, the board contains a new defect or an inaccurate case. In either situation, the case base must be updated to reflect this latest experience. We remark that the general case-based approach goes beyond case matching and database updating functions to possibly include reasoning and extrapolation. Our implementation does not exploit these additional capabilities. Figure 2 shows the complete flow chart for our case-based circuit diagnosis support system (CDSS). We first explian how the system operates before describing how to update the case base (Section 3.4). Section 4 describes enhancements to improve the system's test sequencing capabilities, and Section 5 discusses how to integrate this system with a model-based approach.

At each stage of the search process, the system maintains: (i) a *candidate defect set* consisting of all known defects whose signatures match the observed outcomes for all the tests performed thus far, and (ii) the *available test set* containing all tests that have not yet been performed but whose outcome differentiates one or more candidate defects from the remaining defects in the candidate set. In the basic version of the case-based system, the analyzer chooses a test from the available test set, performs this test, and observes the outcome of the test. (In Section 4 we describe system enhancements to optimally select the next test). When the analyzer enters the observed outcome into the system, the candidate defect set and available test set are updated as follows:

- remove from the candidate defect set all defects whose expected outcome for the latest test differs from the observed outcome. Note that we retain all defects whose outcome for this test is unknown or indeterminate;

- delete the test that was just performed from the available test set; also, remove any test whose outcome is the same (or unknown or indeterminate) for all the remaining defects in the updated candidate defect set.

If the initial case base is comprehensive (i.e., contains all possible defect types, and has enough information regarding expected outcomes to uniquely identify each defect) and accurate, the process must terminate with the candidate defect set containing only the true defect.

To illustrate this process, consider the simple 5-component, series network shown in Figure 3. Suppose we have a board whose abnormal output signal for the functional test indicates malfunction in one of the 5 components (all the intermediate connections on the board are good); for i = 1,2,...,5, defect i refers to a malfunction in component i. We have four available tests that apply a stimulus at the input terminal, and measure the voltage at each of the four intermediate connections (see Figure 3). An outcome of 0, corresponding to abnormal voltage, indicates malfunction of one of the upstream components. Thus, we can represent the signature for each defect as a vector containing 4 binary elements. Defect 1 has the signature {0 0 0 0}, defect 2 has signature {1 0 0 0}, and so on. Table 1 shows the case base containing the complete signatures for all 5 defects. This example does not have any indeterminate outcomes, and we initially assume complete information (i.e., no "unknown" outcomes).

For simplicity, assume that the technician always selects the tests in reverse sequence, i.e., he first applies test 4 (i.e., measures the voltage between components 4 and 5), then test 3, and so on until the defect is conclusively identified. Suppose a board contains defect 3 (i.e., component 3 is defective). Table 2 traces the successive states of the system, i.e., the candidate defect set and the available test set at each stage; the columns of this table answer the defect identification and test selection questions of Section 2.

At the end of the third iteration, the candidate defect set contains a single defect (defect 3), proving conclusively that component 3 on the board is defective. For this example, the chosen test sequence eliminates from the available test set only one test (the test that was just applied) at each step. However, if we had first applied test 3 instead of test 4, its "0" outcome (for this example) automatically eliminates test 4 since all remaining candidate defects {1, 2, 3} have the same outcome "0" for test 4.

This example motivates the following three important observations:

1. *Exploiting prior information*:
   Suppose past history or a priori information (e.g., knowledge about vendor quality problems) suggests that the likelihood of defect 3 is significantly higher than the other defects. Since tests 2 and 3 are adequate to isolate defect 3, the analyzer might choose to

perform these two tests first. For our sample board containing defect 3, this strategy would require only two iterations to complete the diagnosis (compared to the 3 iterations required for the static reverse-order sequence). For complex circuits, this savings in number of tests can be significant. Section 4 discusses how to exploit historical and other prior knowledge to dynamically select an effective test sequence.

2. *Incomplete knowledge*:
Our example assumes that we know the complete signature for each defect, i.e., we know the expected outcome of each test for every defect. However, the partial signature {0 U U U} is sufficient to isolate defect 1 (the element "U" denotes an unknown outcome) if we know that all other defects produce a "1" outcome for test 1; similarly, {U 1 0 U} isolates defect 3, i.e., tests 2 and 3 are the only essential tests to uniquely identify defect 3. Table 3 shows an "incomplete" case base containing adequate knowledge to distinguish between the 5 defects.

Using this case base and our original reverse test sequence, the set of candidate defects for the first 3 iterations remain the same as before (shown in Table 2). However, observing a "1" outcome for test 2 during the third iteration eliminates defect 2 but does not eliminate defect 1 (since the outcome of test 2 for defect 1 is unknown). Therefore, at the start of the fourth iteration, the candidate defect set contains two defects (defects 1 and 3), and we have one available test (test 1) that can eliminate defect 1. We must necessarily perform test 1 (in this case, we will observe a "1" outcome, eliminating defect 1) in order to conclusively prove that the board contains defect 3. Thus, complete knowledge about the expected outcomes for each defect can accelerate the pruning process and reduce the number of tests. Also note that, even though we started with only the partial signatures, we can improve our knowledge after this diagnostic exercise. In particular, we can change defect 1's expected outcome for test 2 from "U" to "0" in the case base, potentially improving future performance.

3. *Multiple defects*:
The signature of a defect represents the anticipated test outcomes assuming that the board contains only that defect. What happens if a board contains multiple defects? Suppose, in our example, the board contained defects 3 and 1. The diagnosis, illustrated in Table 2, is still correct, i.e., component 3 is defective. However, after we verify defect 3 and replace component 3, we must again apply the functional test to confirm proper functioning of the board. If the board malfunctions, the diagnosis process must be repeated; in this example, the second round will detect defect 1.

Although our example used a very simple circuit, it has served to illustrate several generic concepts relating to effective diagnosis. Real circuits contain many complexities such as feedback loops, bidirectional signals, tests that probe the same location but with multiple

measurements, stimuli, and component settings, and so on. Our methodology applies even to these complex circuits.

3.4 Updating the case base: Incomplete knowledge and inaccuracies

We have already seen one opportunity to update the case base and improve subsequent performance: suppose we identify defect i at the end of the diagnosis process, and suppose the current case base contains only the partial signature for defect i. Then, for every test j performed during the current diagnosis whose outcome for defect i was previously unknown, we can replace the unknown entry in the case base with the actual observed outcome for the current board (assuming test j is known to have a deterministic outcome for defect i). We will refer to this updating step as ***signature augmentation***. Note that this updating step is not valid if the board contains other defects besides defect i; hence, we can augment defect i's signature only after repairing it and verifying the proper functioning of the board.

Our discussion in Section 3.4 assumed that the case base is comprehensive (i.e., it covers all possible defect types) and accurate. If the case base is incomplete or inaccurate, the matching and successive defect/test elimination process might not necessarily terminate with a single (true) defect in the candidate defect set. There are two possibilities:

(i) The process terminates because the candidate defect set is empty, i.e., the combination of observed test outcomes does not match with the signature of any known defect. This situation could arise either because:
   (a) the board contains a new defect that had not occured previously;
   (b) at least one of the defect signatures recorded in the case base is inaccurate (due to data entry errors, design changes, etc.); or,
   (c) the observation and measurement of the actual test results was erroneous.

   In all three cases, the board requires "extraordinary" actions to identify the true symptoms and cause of the problem; these actions might entail detailed probing and experimentation by the analyzer as well as consultations with design and test engineers, production staff, and component vendors. If the board is found to contain a known defect, its signature in the case base must be checked and corrected, if necessary. We refer to this updating step as ***signature correction***. On the other hand, if the board contains a new defect, we must add a new case (row) corresponding to that defect. We refer to this step as ***case addition***. The new case contains the expected test outcomes for that defect, and possibly some auxiliary information such as the name of the defect, the analyzer or engineer who diagnosed the defect or revised the signature, the corrective (repair) action required, the defect category, and who to notify (or how to use) for quality improvement. Although the existing tests suffice to distinguish this

new defect from the previously known defects (the observed test outcomes that led to the empty defect set provide adequate "proof" for this defect), the analyzer might wish to add one or more new tests that can quickly isolate this defect.

(ii) The diagnosis process terminates because the available test set is empty, but the candidate defect set contains more than one possible defect. This occurrence indicates erroneous test measurements, inaccurate defect signatures, or inadequate tests. "Inadequate tests" refers to the situation when the current tests cannot distinguish between the candidate defects either because many test outcomes are unknown or new tests are required. Recall that we delete a test from the available test set when we perform that test or if its expected outcome is the same (same known value, indeterminate or unknown) for all the remaining candidate defects. Again, this stopping condition (with multiple candidate defects and no remaining tests) requires extraordinary actions to verify the test measurements, check the defect signatures stored in the case base, complete the partial signatures of the remaining candidate defects, or add new tests that distinguish between these defects. We refer to this process as ***test addition.***

Figure 2 contains a flow chart summarizing the case matching and updating process. This process includes a *defect verification* step at the end of a successful diagnosis exercise. Verification entails confirming previous observed outcomes and applying any tests that remain in the available test set; if one or more outcomes conflict with the hypothesized defect, a new case must be added. This optional step is especially important after design changes and during production ramp-up when many new cases are added to the case base. During normal operation, defects are automatically verified if, after repairing the identified defect, the board passes the functional test. A board failing the test again requires explicit defect verification to determine whether the original diagnosis was erroneous or if the board contains multiple defects.

In summary, the case-based approach to support circuit diagnosis permits *incremental, robust, and distributed knowledge acquisition*. Unlike other knowledge-based approaches, the case-based system employs on-line and incremental, rather than batch-oriented, knowledge acquisition. All cases have the same format; a case contains information on the symptoms that the user identifies to be relevant in defining and distinguishing between defects. Because of the standardized format, the case base is easy to understand and update. The system begins with incomplete knowledge and poor defect coverage, but each unsuccessful diagnosis initiates a process of case-building to ensure subsequent coverage of the same defect. Our experience suggests that the case coverage increases very rapidly during the first few weeks. By only maintaining the diagnostic knowledge that is needed, the system can provide quick response. Incompleteness and inconsistencies in the case

base are easy to detect: they result in unusual termination of the diagnosis process (empty candidate defect set or empty available test set). The various correction and updating steps—signature augmentation, signature correction, case addition, and test addition—make the knowledge acquisition process robust and adaptive. The case-based system can be implemented using a shared relational database in a multi-user computing environment, thus providing wide access to analyzers at different locations as well as design, test and process engineers and production supervisors. This implementation also allows the case base to be integrated with other factory information systems, conveying current vendor, design, and process problems to the diagnosis operation, and feeding back defect information for quality improvement. The system can also keep track of the cumulative relative frequencies and trends of different defects; we next describe how to use these relative frequencies to guide and optimize the sequential testing process.

4. Optimal Test Selection

By automating the search and updating functions, the basic CDSS reduces the time for each iteration of the diagnosis process. Decreasing the number of iterations provides another means to significantly reduce the total diagnosis time per board. As our example of Section 3 illustrates, the number of iterations depends on the test sequence. We, therefore, require an effective test selection policy or rule: at each stage of the diagnosis process, the method helps us decide which available test to apply next (based on the observed outcomes thus far). We refer to a test selection rule that minimizes the expected number of tests as the *optimal test selection policy*. This section presents a dynamic programming formulation for the optimal test selection problems, and describes on-line heuristic rules that the basic CDSS can incorporate to reduce the expected number of tests. These rules rely on prior information concerning the likelihood of different defect types, estimated from historical data on defects (relative frequencies and trends), knowledge about specific problems at the vendor site or in the assembly process, inputs from design engineers on new engineering changes, etc.

4.1 Test selection example

We first consider a simple example to better understand the impact of test structure and sequencing on the number of tests required to prove the defect. Suppose a board contains one of n potential defects, and each defect has a single associated test that is both necessary and sufficient to "prove" that defect, i.e., test i has a "1" outcome for defect i, and a "0" outcome for all other defects (or vice versa). We will refer to this type of test as a *focused test* (test 1 in Figure 3 is a focused test). Thus, the complete case base for n focused tests corresponds to a n x n identity matrix. Suppose n = 5, and the five possible defects have the following probabilities of occurence: 0.1, 0.1, 0.2, 0.2, and 0.4. Intuitively, applying the tests in order of decreasing defect probabilities minimizes the expected number of tests.

For our example, the "decreasing probability sequence" corresponds to applying the tests in reverse index order (i.e., 5-4-3-2-1) until the defect is identified. The expected number of tests, say E_1, using this strategy is:

$$E_1 = 1x0.4 + 2x0.2 + 3x0.2 + 4x0.1 + 5x0.1$$
$$= 2.3 \text{ tests per board.}$$

On the other hand, for a static policy that applies tests in the forward sequence 1-2-3-4-5, the expected number of tests E_2 is:

$$E_2 = 1x0.1 + 2x0.1 + 3x0.2 + 4x0.2 + 5x0.4$$
$$= 3.7 \text{ tests per board.}$$

The optimal strategy saves, on average, 1.4 tests per board relative to the second sequence, a 43% savings. (A more skewed probability distribution can give even greater savings.) Thus, optimal test sequencing has considerable potential to reduce diagnosis cost, and provide quick feedback for quality improvement.

Because we assumed that each defect has a focused test, our example has a simple optimal testing policy that is static (i.e., does not vary with the observed outcomes) and easy to implement, i.e., sequence the tests in decreasing order of defect probabilities. Since focused tests are infeasible for certain defects (because of constraints on probing, measurement, etc.) and eliminate only one defect at a time (except at the final step), analyzers rely on a combination of focused and *joint* tests. Unlike a focused test that is both necessary and sufficient, joint tests cannot isolate defects individually but their combined outcomes can jointly identify and prove the defect (e.g., tests 2 through 4 in Figure 3). Joint tests reduce the number of required tests to diagnose all defects, and can, under certain circumstances (depending on their structure and the relative likelihood of various defects), reduce the expected number of iterations per board. The "test configuration problem" of deciding what combination of focused and joint tests to include in the available test set poses challenging tradeoffs between the number of available tests and the expected number of iterations per board; we do not address this tradeoff in this paper, assuming instead that the available tests are predetermined. When the available test set contains joint tests, the optimal test selection policy might be dynamic, i.e., the decision on what test to apply next depends on the current "state" of the system defined by the previous test outcomes; we next formulate this test selection problem.

4.2 Test selection: Problem formulation and optimal solution

The *test selection problem* seeks an optimal or near-optimal testing *policy* that minimizes the expected number of tests or iterations to diagnose a malfunctioning board. We distinguish between dynamic and static policies; a dynamic policy uses information regarding previous test outcomes to select the next test, while a static policy selects the same sequence regardless of the test outcomes. Dynamic policies perform better (i.e., they require fewer tests on average) since they use more information. The optimization of

sequential search processes has been previously modeled (see, for example, Whinston and Moore [1986]), and applied, for instance, to computer file search (Moore, Richmond, and Whinston [1988]). The reliability literature has also addressed optimal test sequencing issues for certain special circuit and test structures (e.g., Nachlas, Loney, and Binney [1990], Butler and Lieberman [1984] study series systems with focused tests). However, unlike our dynamic knowledge acquisition context, this stream of literature often assumes that all defect types and their probabilities are known, and the requisite tests are available, without provisions for unknown or indeterminate outcomes. Furthermore, the knowledge-based diagnosis systems described in the literature do not appear to incorporate optimal search principles.

4.2.1 Dynamic Programming Formulation

This section presents a standard dynamic programming formulation to clarify the ingredients and tradeoffs in test selection during circuit diagnosis. First, we introduce some notation. We use the indices i = 1,2,...,n and j = 1,2,...,m to represent the n defect types and m available tests. We assume, without loss of generality, that the case base contains all possible defect types. Otherwise, we can introduce a dummy defect type called "unidentified" (which includes "no trouble found"), and a fictitious focused test with a "1" outcome if the board has an unidentified defect, and a "0" outcome for all other known defect types (we perform this fictitious test only when the set of known candidate defects is empty). We permit incomplete knowledge of defect signatures. However, the known partial signature for each defect must be accurate, and we must have enough information (i.e., the analyzer must known the expected deterministic outcomes for enough defect-test combinations) to unambiguously identify the true defect. We also assume that the test observations and measurements are error-free. These assumptions ensure that the diagnosis process is accurate, and always terminates "properly" with the candidate defect set containing only the true defect.

At any intermediate iteration of the sequential search process, the observed outcomes for the tests we have performed thus far determine the remaining candidate defects (i.e., defects whose expected outcomes are consistent with the observed outcomes) and available tests (that can distinguish between the remaining defacts); hence, the previously observed outcomes completely capture our current knowledge about the board. We let X denote the *state vector* representing this current knowledge. We specify the state X as the following m-vector (where m is the number of initially available tests): the j^{th} element x_j can take three values: $x_j = 0$ if test j was previously performed and its observed outcome was "0", $x_j = 1$ if test j's outcome was "1", and $x_j = N$ if test j has not yet been applied. The initial state vector X^0 has elements $x_j^0 = N$ for all j = 1,2,...,m. Let D(X) and T(X) denote, respectively, the index sets of candidate defects and available tests corresponding to any state X. Initially, $D(X^0) = \{1,2,...,n\}$ is the set of all possible defects, and $T(X^0) =$

{1,2,...,m} is the set of all available tests. Starting with this initial state, the iterations of the diagnosis process correspond to moving from one state to the next until we reach a *terminal* state X^* whose candidate defect set $D(X^*)$ contains only one defect. At any intermediate state X, we select and apply a test $j \in T(X)$, observe its outcome O_j (= 0 or 1), and update the state vector (change x_j from N to O_j).

The expected number of tests to complete the diagnosis of a defective board depends on the probability of different defects, and the testing policy that we choose. For the remainder of this section, we will assume that the board contains only one defect. Let $p_i^0 \geq 0$ denote the *prior probability* (at the start of the diagnosis process) that the board contains defect i. For instance, we might set p_i^0 equal to the relative frequency of defect i during the previous month; alternatively, we might subjectively estimate this prior probability based on qualitative information (e.g., new operators on the line, problems at the vendors' facility). Since, by our definitions and assumptions, the n defect types are mutually exclusive and collectively exhaustive, the prior probabilities sum to 1, i.e.,

$$\sum_{i=1}^{n} p_i^0 = 1. \qquad (4.1)$$

As we apply tests and eliminate defects from the candidate defect set, we must update the defect probabilities to reflect the test outcomes. Let $p_i(X)$ represent the conditional probability that the board contains defect i given that we are currently in state X. Any defect i that does not belong to the candidate defect set D(X) must have zero conditional probability, and the conditional probabilities for all remaining defects must sum to 1. Thus,

$$\begin{aligned} p_i(X) &= 0 && \text{if } i \notin D(X), \text{ and} \\ &= p_i^0 / \{ \sum_{h \in D(X)} p_h^0 \} && \text{if } i \in D(X). \end{aligned} \qquad (4.2)$$

At any intermediate (non-terminal) stage of the diagnosis process, our choice of the next test only depends on the current state; the exact sequence of previous tests that led to the current state is irrelevant. Therefore, if we are currently at state X, the optimal test selection policy must choose the test $j \in T(X)$ that minimizes the expected number of remaining tests until the diagnosis process terminates. These features suggest the following dynamic programming formulation for the test sequencing problem. We define the *minimum cost-to-go* $C^*(X)$ from state X as the expected number of remaining tests if, at each subsequent state, we select the next test "optimally". Let $j^*(X) \in T(X)$ represent the index of the "optimal" test to apply if we are in state X. Let us describe how to identify this optimal test.

For each test $j \in T(X)$, let $C(X,j)$ denote the cost-to-go from the current state X if we perform test j. Applying test j gives two possible outcomes $O_j = 0$ or 1, each leading to a different next state (obtained by changing x_j from its current value N in state X to 0 or 1, respectively). Each of these two next states has an associated minimum cost-to-go. We can, therefore, express $C(X,j)$ in terms of these subsequent minimum costs-to-go. For this purpose, let us define $\pi(X,O_j)$, for $O_j = 0$ or 1, as the probability that applying test $j \in T(X)$ at state X results in outcome O_j. This probability depends on the conditional probabilities of the defects in D(X), and their expected outcomes. Let D0, D1, DU, and DI represent the respective subsets of defects in D(X) whose expected outcomes (recorded in the case base) for test j are 0, 1, unknown, and indeterminate (these subsets depend on the current state X and the test j; we omit the arguments X and j for simplicity). Without additional information, we assume that any defect whose outcome for test j is unknown or indeterminate is equally likely to give a "0" or "1" outcome. Thus, $\pi(X,O_j= 0)$, the probability that test j will give a "0" outcome when it is applied at state X is:

$$\pi(X,O_j=0) \;=\; \sum_{i \in D0} p_i(X) \;+\; 0.5 * \sum_{i \in DI \cup DU} p_i(X). \qquad (4.3)$$

Similarly, the likelihood of a "1" outcome for test j applied at state X is:

$$\pi(X,O_j=1) \;=\; \sum_{i \in D1} p_i(X) \;+\; 0.5 * \sum_{i \in DI \cup DU} p_i(X). \qquad (4.4)$$

Using these probabilities, we can compute the cost-to-go $C(X,j)$ as follows:

$$C(X,j) \;=\; 1 + \pi(X,O_j=0)\; C^*(X \cup \{O_j=0\}) + \pi(X,O_j=1)\; C^*(X \cup \{O_j=1\}), \qquad (4.5)$$

where $X \cup \{O_j=0\}$ and $X \cup \{O_j=1\}$ denote, respectively, the next states (with $x_j = 0$ or 1) when the outcome O_j of test $j \in T(X)$ is 0 or 1. The first term (i.e., the constant 1) on the right-hand side of equation (4.5) represents the "cost" of applying test j (recall that our cost function counts the number of tests that are performed), and the last two terms contain the minimum cost-to-go from the two possible next states. We then compute the minimum cost-to-go $C^*(X)$ from state X as the minimum value of $C(X,j)$ over all available tests $j \in T(X)$; the test that achieves this minimum is the optimal decision $j^*(X)$, i.e.,

$$C^*(X) \;=\; \text{Min } \{\, C(X,j): j \in T(X) \,\}, \text{ and} \qquad (4.6)$$

$$j^*(X) \;=\; \underset{j \in T(X)}{\text{argmin}}\; C(X,j). \qquad (4.7)$$

If X is a terminal state, i.e., D(X) contains exactly one defect, then we do not require any additional tests; hence, the minimum cost-to-go

$$C^*(X) \;=\; 0 \qquad \text{if } |D(X)| = 1, \qquad (4.8)$$

where |S| represents the number of elements in set S. Equations (4.6) and (4.8) are the dynamic programming recursive equations to compute the minimum cost-to-go and the optimal decision for every state X.

Dynamic Programming Procedure:

- Initialization:

  For every terminal state X, set $C^*(X) = 0$.

  Initialize: Labeled_states ← set of all terminal states.

- Iterative step:

  For every state X whose next states all belong to Labeled_states,

  use equation (4.5) to compute C(X,j) for every test $j \in T(X)$;

  use equation (4.6) to compute $C^*(X)$;

  equation (5) gives the optimal decision $j^*(X)$ at state X;

  add state X to Labeled_states;

  Repeat iterative step until all states are labeled.

At every iterative step of this solution procedure, we can find at least one "unlabeled" state whose next states $X \cup \{O_j=0\}$ and $X \cup \{O_j=1\}$, for every next test $j \in T(X)$, are all labeled (otherwise, the state transition diagram must contain a directed cycle, which is impossible). The dynamic programming formulation can easily accomodate other objective functions (e.g., minimize the expected total testing "cost" when tests have different costs) as well as precedence constraints (e.g., test j must precede test j' due to technological or operational constraints) and probabilistic test outcomes.

We can represent the optimal testing policy found by the dynamic programming algorithm as a "look-up" table; this table specifies the best next test $j^*(X)$ for every possible state X. Let us illustrate how the analyzer might use this table during diagnosis. Let X^k represent the state at the end of the k^{th} iteration of the diagnosis process, and suppose X^k is not a terminal state. At iteration (k+1), we use the look-up table to identify the optimal next test $j^* \in T(X^k)$ for state X^k, apply this test, and observe its outcome O_{j^*}. We then update the state vector by setting the j^{th} element equal to the observed outcome, i.e.,

$$\begin{aligned} x_j^{k+1} &= x_1^k \quad \text{if } j \neq j^*, \text{ and} \\ &= O_{j^*} \quad \text{if } j = j^*. \end{aligned} \tag{4.9}$$

Eliminating the candidate defects that have an expected (known) outcome different from O_{j^*} gives the new, smaller candidate defect set $D(X^{k+1})$ corresponding to the new state X^{k+1}. If $D(X^{k+1})$ contains only one defect, the diagnosis process is complete. Otherwise, we repeat the above process for the new state. The dynamic program can be solved "off-line" to create the decisition table. By storing this table in a computer database, the case-based diagnosis support system can automate the look-up and test selection functions.

Although the dynamic programming procedure is easy to state and can be solved off-line, it is computationally intensive for complex circuits that have a large number of available tests. Furthermore, we must re-solve the problem every time the prior probabilities p_i^0 change. We, therefore, consider some on-line *heuristic* rules to select the next test at each iteration of the diagnosis process.

4.3 Heuristic Decision Rules

Easiest to implement are on-line rules that dynamically select the next test at each stage (instead of referring to a look-up table) based on simple computations using information about the current state. Intuitively, the on-line rule must select a test that can rapidly prune the candidate defect set. We list below three out of many possible rules to achieve this objective.

Highest failure rule:

This rule selects the test with the highest probability of failure, where we define the least probable outcome of a test as its "failure" mode. This rule is patterned after the optimal strategy for our example in Section 4.1 where we always select the "focused" test that proves the most likely candidate defect (the test proves the defect by failing).

Most known outcomes rule:

Recall that we permit indeterminate outcomes as well as incomplete knowledge of certain (deterministic) test outcomes. A test with unknown or indeterminate outcomes for many defects is relatively ineffective since these defects remain in the candidate set regardless of the test's actual outcome. Hence, to rapidly prune the candidate defect set, this rule selects the available test with the largest number of known outcomes (for the remaining candidate defects). We might enhance this rule by incorporating defect probabilities (e.g., we prefer a test for which the sum of probabilities of all candidate defects with unknown or indeterminate outcomes is the smallest).

Smallest expected remaining defects rule:

To rapidly home in on the true defect, this rule selects the available test j with the smallest expected number of remaining defects (which is the sum, over the two outcomes $O_j = 0$ and 1, of the probability of obtaining outcome O_j times the number of remaining defects if we observe outcome O_j). Observe that a test that has many unknown or indeterminate outcomes (see previous rule) will likely have a large number of remaining defects for either outcome. Again, we can modify this rule to account for the "weight" or probability of the remaining defects.

We might enhance these myopic rules to incorporate "look-ahead" features, i.e., to consider the likely states two or more iterations hence. Alternatively, we might consider "static" rules that priortize the tests before starting the diagnosis process; at every stage, the

available test with the highest priority is applied next. Unlike the dynamic programming procedure, these heuristic rules do not guarantee optimality, i.e., in the long run, they might require more tests on average than the optimal policy. Their relative effectiveness depends on the nature and probability distribution of various defects, the test characteristics (e.g., focussed versus joint tests), and the level of knowledge regarding test outcomes.

To study the relative effectiveness of heuristic rules, Semmelbauer [1992] conducted an extensive simulation study. The study applied selected heuristic rules to many random problem instances with varying characteristics such as the number of possible defect types, the number of available tests, the "density" of the case base (i.e., the fraction of known test results), and the shape of the defect probability distribution (e.g., a skewed distribution corresponds to a few commonly occuring defects and many "rare" defects, while an uniform distribution implies approximately equal likelihood for all defects). For each diagnosis environment, the simulation generates several problem instances (i.e., defective boards) and records the number of tests required to complete the diagnosis using each heuristic rule. The average number of tests over these random instances estimates the expected number of tests for each rule. A naive rule that randomly selects the next test at each stage provides a benchmark for comparison. We summarize some important conclusions from these simulations (for more details, see Semmelbauer [1992]):

- Principled test sequencing can give significant savings: Even relatively simple heuristic rules (such as the most known outcomes rule) gave savings of approximately 30% (in the number of tests performed) relative to random test sequencing.
- The number of tests performed decreases as the density of the case base increases. A denser case base (i.e., fewer unknown and indeterminate outcomes) eliminates more defects at each stage since it contains more information.
- Problem size (number of defect types) and the skewness of the distribution of defect probabilities do not appear to have a significant impact on the relative performance of different heuristics.

Next, we discuss some lessons we learned by implementing a prototype of the case-based circuit diagnosis system.

5. Prototype Implementation and Future Directions

5.1 Features of CDSS prototype

We implemented and tested a prototype of the Circuit Diagnosis Support System (CDSS) in a production environment to diagnose an actual printed circuit board. The primary purpose of the implementation was to prove the concept, and understand human issues associated with introducing computer support tools in a predominantly manual

diagnosis environment. The prototype was written in Omnis 5 for Macintosh SE computers at the technician stations; these terminals use touch-screens, bar-code readers, mice, and keyboards as input devices. The Omnis program communicates via an SQL (Standard Query Language) interface with a relational database (ORACLE) on the central UNIX machine serving the facility. This central database stores and manipulates the case base, permitting multi-user access via a local network. The prototype system was not integrated with the factory information system, and used only a simple decision rule (the most known outcomes rule) to select the next test at each stage. Also, the prototype did not incorporate the signature augmentation feature described in Section 3. Semmelbauer [1992] describes the prototype implementation in greater detail.

Observations and feedback from users over a five-week period led to several system improvements. The final version of the system employs a user-friendly interface, shown in Figure 4, that displays matching cases (the candidate defect set), and previous tests and outcomes, and permits technicians to assess the usefulness of each available test in pruning the candidate defect set. One of the important system features is an user-specified option to override the automatic test selection procedure; using this option, technicians can choose the next test manually from the list of available tests. The simplicity of the case-based approach, its transparent logic and operation, and the manual override option greatly facilitated the introduction and acceptance of the system in the existing environment.

The system was tested on a double-sided board containing approximately 400 components, half analog (radio frequency) and half digital. Only part of the board was supported by the CDSS; the test circuit contains approximately 70 analog components. Over the course of five weeks, 467 boards were diagnosed using the prototype system; these boards had 87 different defect types (involving 54 components, some of which had multiple defect types). We estimate that the average time to diagnose each board dropped from about 25 minutes per board for purely manual diagnosis to approximately 10 minutes with CDSS assistance. (This estimate is based on an informal observation for random boards rather than a rigorous time study.) One potential disadvantage of the case-based approach is the degradation in performance if the case base contains a very large number of cases to be searched. In practice, we found that the response time was not adversely affected even after five weeks of case addition. For situations that require a significantly higher number of cases, we might consider limiting the number of cases that are stored: when the case base reaches this limit, an old case (with the smallest current probability of occurence) must be discarded in order to add a new case. This strategy trades off the gain in response time per test iteration against the additional effort needed to re-learn and update the case base if an old defect occurs again.

To evaluate the system's learning rate, we measured the *coverage* of the system during each week of the trial period. Coverage is defined as the proportion of defects occuring during a week that is correctly diagnosed using the current information in the case base. Figure 5 shows how the system's coverage increased during the first five weeks of its operation. The case base initially contained cases corresponding to defects that the technicians felt were most likely or significant (based on their experience). As Figure 5 shows, after 3 weeks of using the CDSS, 85% coverage was achieved for the test circuit; by the fifth week, coverage increased to 90%. Interestingly, during the first week of operation, the system could only diagnose about 17% of the defect types, suggesting that initial knowledge can be quite inadequate for this application. The CDSS's rapid learning rate makes it especially attractive in today's dynamic electronics assembly environment.

5.2 Managing the diagnosis function

Although the CDSS is effective and easy to use, exploiting its full potential requires appropriate management support. To rapidly develop full defect coverage, the system relies on its regular use by experienced technicians as they detect and diagnose new defects. However, as with many knowledge-based systems, experienced technicians are less likely to use the system because they believe that they have the least to learn. Overcoming this phenomenon requires some fundamental changes in the way the diagnosis function is managed.

Technicians have traditionally been managed as hourly production workers despite their important role in the quality improvement loop and their high skill level. Emphasizing their contributions to defect prevention and process improvement, rather than viewing diagnosis as their sole function, can produce rapid progress towards world-class manufacturing standards. This shift in emphasis from fault finding to proactive process improvement requires changing the technicians' performance evaluation metrics, providing appropriate incentives, and creating an environment that is conducive to learning and information sharing. Based on our experience with the prototype system, we suggest the following initial steps to implement these changes and exploit the technicians' skills:

1. Change the technician's job description and expectations: Technicians' first priority must be defect prevention and process improvement. They are among the most skilled workers on the line, and are often the first to identify defects. Thus, they are better suited than the design or process engineers (who are typically less concerned with day-to-day operations) to respond quickly to production problems.

2. Measure quality as well as quantity: Emphasizing the quality of diagnosis and the consequent improvements and learning can create a productive culture. Traditional productivity metrics such as the number of boards diagnosed per month often increase the variability in diagnosis times by prompting technicians to work on the "easy" boards first.

3. Introduce technical supervision through master technician. Traditionally, technicians report to the production or line supervisor who often does not have detailed technical knowledge about circuit behavior. Appointing a master-technician to supervise the diagnosis/repair operation and provide consulting help to the technicians can greatly improve productivity, especially in an environment containing multiple assembly lines.

4. Interact closely with product development. Closer cooperation with product development might be encouraged, for instance, through a rotation program or a career path from technician to development engineer.

We believe that electronics companies will be naturally driven towards these changes in order to cope with increasing product diversity and small-lot production.

5.3 Future directions

Our prototype implementation can be easily enhanced to incorporate several system features that we described in Sections 3 and 4. These enhancements include: (i) permitting tests to have more than two outcomes, (ii) providing defect validation and verification as an explicit feature, (iii) incorporating the automatic case base updating mechanisms (e.g., signature augmentation), (iv) integrating fully with existing factory control and information systems, (v) collecting actual test measurements (continuous values) and automatically inferring the corresponding test outcomes, (vi) permitting more than one defect per board, (vii) minimizing the expected testing cost per board when tests have different costs, (viii) accounting for precedence constraints and sequence-dependency among tests, (ix) incorporating more sophisticated test sequencing methods based on defect probabilities, (x) permitting multiple signatures for the same defect, and (xi) adjusting for error-prone measurements (i.e., permitting tests with false positive and false negative outcomes; see, for example, Nachlas et al. [1990]). These enhancements are possible even with current hardware, software, and development tools. The case-based approach offers two very promising possibilities in the future—integrating with a model-based approach, and adding automated probing capabilities.

We have argued that, while the model-based approach offers the promise of seamless and instantaneous design-diagnosis integration, the method is not suitable for real-time applications because of its extensive computational requirements. One very promising approach is to combine the model-based and case-based approaches in a *hybrid system* that captures the automatic reasoning capabilities of diagnostic models with the quick response of a case base that stores only the most important and relevant defects. The model-based approach can complement the case-based system in two ways: (i) whenever a new defect or new test is added to the case base, we can use the circuit model to verify the correctness of the partial signature (specified by the analyzer or obtained by recording the test outcomes which led to the new defect) and even augment it (i.e., replace unknown values in the case base with model-based predictions of test outcomes); and, (ii) when the case-based

approach terminates with an empty candidate defect set, the circuit model can generate hypothesis regarding the new defect, and propose appropriate tests to isolate it. We can also apply the model-based system off-line to initialize the case base. Note that these schemes apply the circuit model only to resolve exceptions; hence, they do not degrade the average response time of the system during regular diagnosis. The hybrid approach, therefore, offers the advantage of a robust case-based system that permits user interaction and provides rapid on-line response, with the enhanced model-based capabilities to react quickly to changes in product design (using model-based reasoning). This decomposition of the diagnosis function into its short-term and long-term components permits independent development of the case-based and model-based subsystems; for example, we might first develop the case-based system using currently available hardware and software, preparing the infrastructure for a future implementation of the model-based component when the state-of-the-art improves to make the approach practicable.

The second promising direction concerns the addition of automated probing capabilities to the CDSS. As printed circuit assemblies become smaller and more dense, automatic testing and probing will become a necessity. Most test equipment, such as power supplies, signal generators, and frequency analyzers, are already controllable by a computer; however, currently most diagnosis operations use manual probing. With the emergence of new robotic technologies with precise and swift movement capabilities (with vision and feedback control), automated flexible probing becomes feasible in manufacturing settings. A case-based system controlling the automatic circuit probing and measurement equipment can reduce diagnosis time by performing routine and repetitive diagnosis functions without human intervention; analyzers would still be required to troubleshoot new defects and resolve inconclusive diagnoses.

In conclusion, this paper has proposed a viable approach to reduce diagnosis time and effort, and provide rapid feedback for quality improvement. A novel feature of our case-based approach is its ability to incorporate principled test selection methods. Although we developed the dynamic programming formulation and heuristic test selection rules in the context of off-line diagnosis, the underlying principles also apply to in-line (in-circuit and functional) testing.

Computer assisted diagnosis is, of course, just one element of the overall testing and repair strategy. We have alluded to the many other decisions and tradeoffs to consider in formulating an effective testing strategy. These decisions include:

- how to incorporate diagnosis and testability issues in the product design process? Williams and Parker [1983] present a comprehensive survey of issues and methods relating to design for testability;
- where to locate in-line test stations, and what tests to perform at each station?

Pappu and Graves [1992] discuss this issue in a generic production line setting, and Chevalier [1992] considers alternative in-line testing strategies for printed circuit board assembly operations;

- whether to perform detailed, diagnostic tests in-line or off-line?
  As we noted in Section 2, this decision depends on the tradeoff between higher cycle times and test capacity requirements on the main assembly line versus greater off-line effort;
- how to "discretize" the continous outcomes of tests?
  For instance, the test measurements in analog circuits are continuous values; the test engineer must specify ranges of values that correspond to, say, "PASS" or "FAIL" outcomes for the test. When the measurements are error-prone, Chevalier [1992] proposes a method to set the ranges based on the relative costs and probabilities of false acceptance and rejection;
- what tests to include in the test set for off-line diagnosis?
  As we discussed in Section 4 the choice of joint versus focused tests impacts the expected number of iterations to complete the diagnosis;
- when to stop the testing process for a board?
  Our discussion implicitly assumed that the defect must be conclusively identified so that the board can be repaired. For some defects, the cost to diagnose and repair the board (including the cost of work in process, etc.) might exceed the value of the board. When boards contain multiple defects, a common policy is to discard any board that goes through the diagnosis-repair loop more than a certain prespecified number of times. Conversely, analyzers might sometimes terminate the diagnosis process prematurely (before conclusively proving a defect), and recommend repairing the most likely cause of the problem; and,
- how to screen boards that enter diagnosis?
  Ou and Wein [1992] and Longtin et al. [1992] discuss screening strategies to discard semiconductor wafers with low yield in order to avoid wasteful usage of the subsequent detailed wafer-probing capacity. Similar screening strategies might alleviate diagnosis bottlenecks in printed circuit board assembly.

As these issues illustrate, testing and diagnosis in printed circuit board assembly operations continues to offer many important and challenging research opportunities. Finally, circuit board diagnosis, as we have defined it, focusses on finding the source of the problem so that the board can be repaired. For process improvement, we need to trace the problem to its root cause, i.e., the component vendor, board design, or processing step responsible for introducing the defect; proactive process control requires a further step, namely, anticipating and preventing defects. Given the massive volumes of data generated by the placement, reflow, inspection, testing, diagnosis, and repair stations in today's high volume electronics assembly environment, we require automated systems that

systematically monitor and screen the signals, recognize patterns, identify the root causes, and initiate process control and improvement activities.

Acknowledgments:
We are very grateful to Brian Santoro, Thomas Babin, and Dennis Miller for initiating this project and providing constant support and encouragement. We thank the analyzers, especially Zakiuddin Mashood and Karl Browder, and members of the DOC team for sharing their insights on circuit diagnosis.

Table 1: Complete Case base for 5-component, serial network

	Test 1	*Test 2*	*Test 3*	*Test 4*
Defect 1	0	0	0	0
Defect 2	1	0	0	0
Defect 3	1	1	0	0
Defect 4	1	1	1	0
Defect 5	1	1	1	1

Table 2: Diagnostic Process for 5-component, serial network

Iteration Number	Candidate Defect Set	Available Test Set	Selected test	Test Outcome
1	{1, 2, 3, 4, 5}	{1, 2, 3, 4}	4	"0"
2	{1, 2, 3, 4}	{1, 2, 3}	3	"0"
3	{1, 2, 3}	{1, 2}	2	"1"
4	{3} STOP			

Table 3: Incomplete Case base for 5-component, serial network

	Test 1	*Test 2*	*Test 3*	*Test 4*
Defect 1	0	**U**	**U**	**U**
Defect 2	1	0	**U**	**U**
Defect 3	1	1	0	**U**
Defect 4	1	1	1	0
Defect 5	1	1	1	1

Figure 1: Cumulative probability of various defects

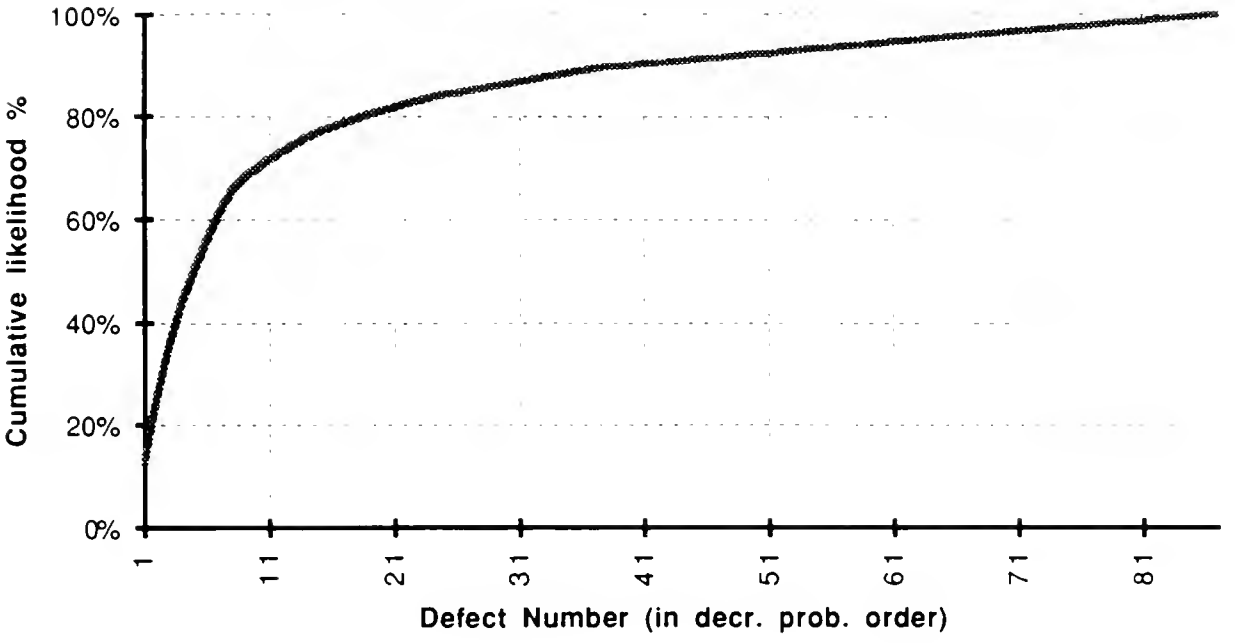


Figure 2: Flow Chart for Case-based CDSS

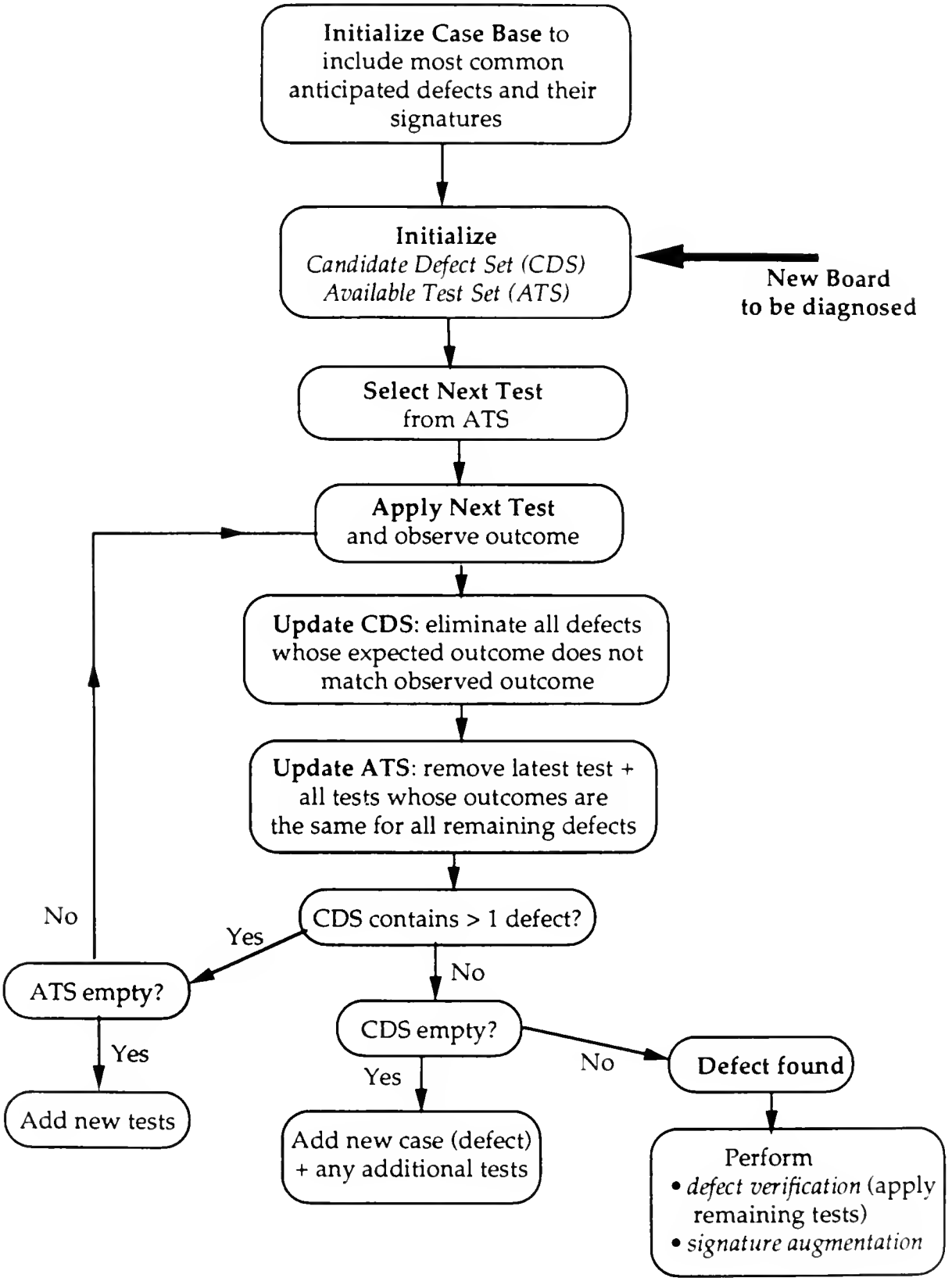
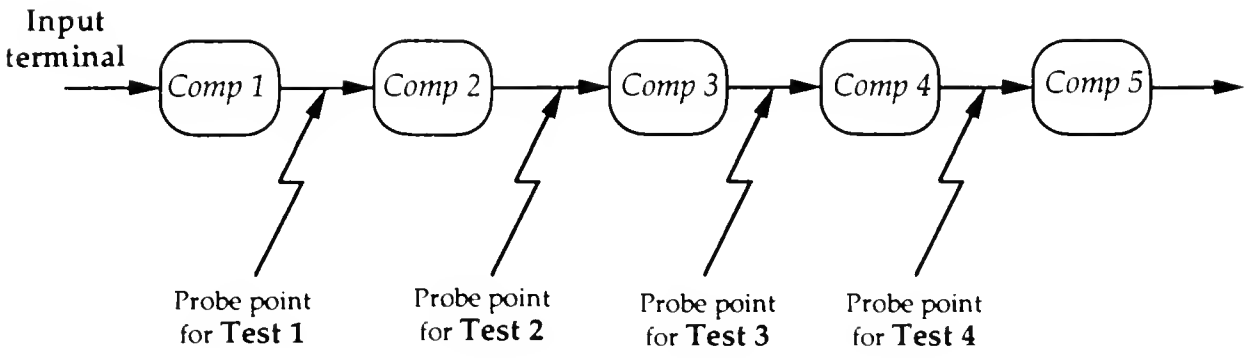


Figure 3: Series network example



Defect i implies component i is defective, for i = 1,2,...,5
Test i measures the voltage between components i and (i+1), for i = 1,2,3, and 4.

Figure 4: Sample screens from CDSS prototype

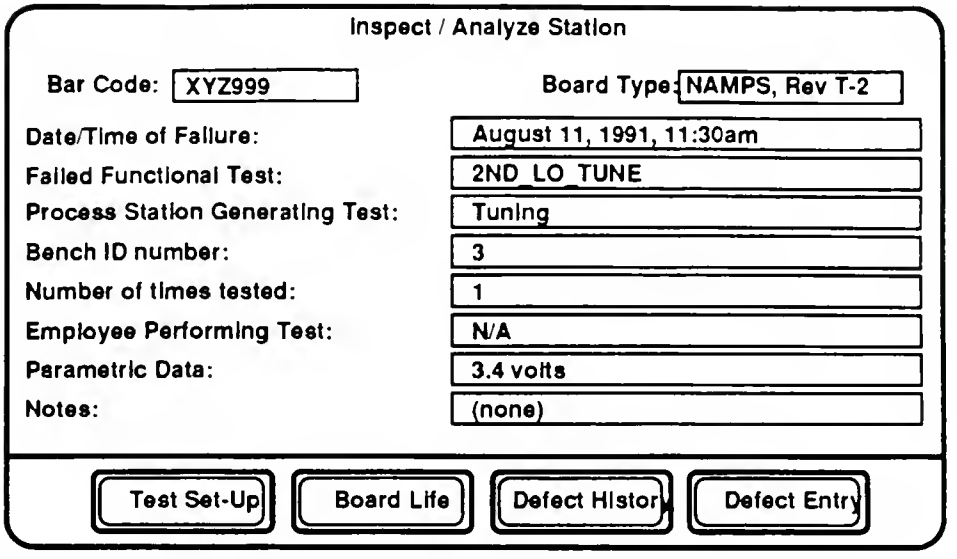


Figure 4a. Screen display after board fails a functional test.

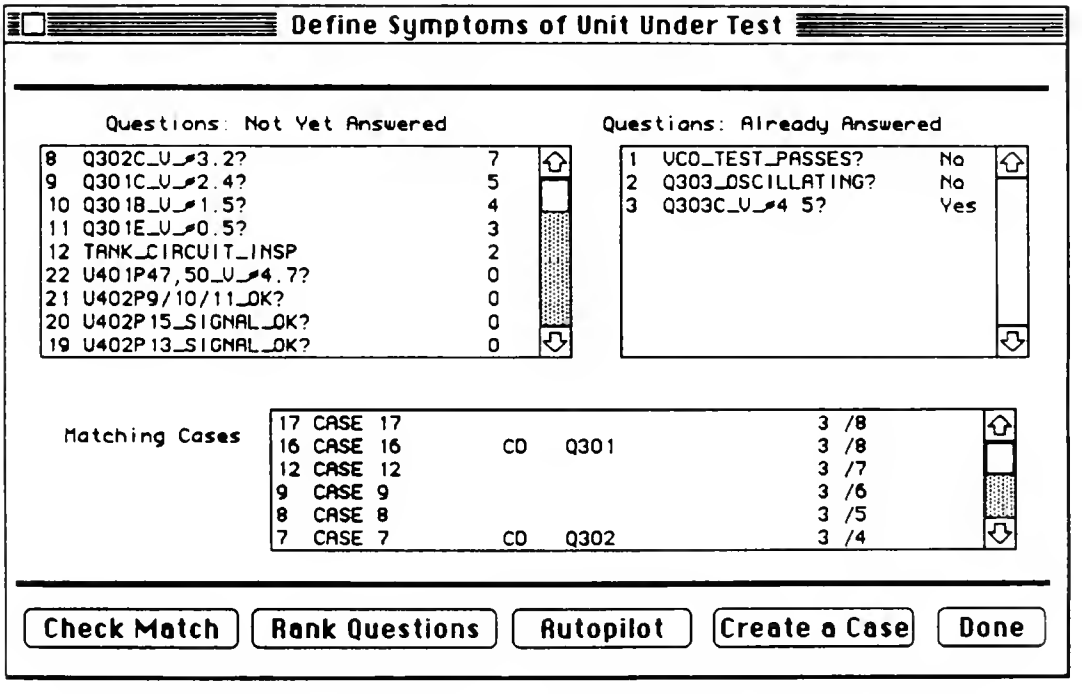


Figure 4b. CDSS prototype in manual mode, after some tests have been performed. Notation "3/8" next to matching case 17 means that 3 of the 8 test-results that are defined in the case match currently. The questions or tests are listed in decreasing order of number of known outcomes for remaining cases

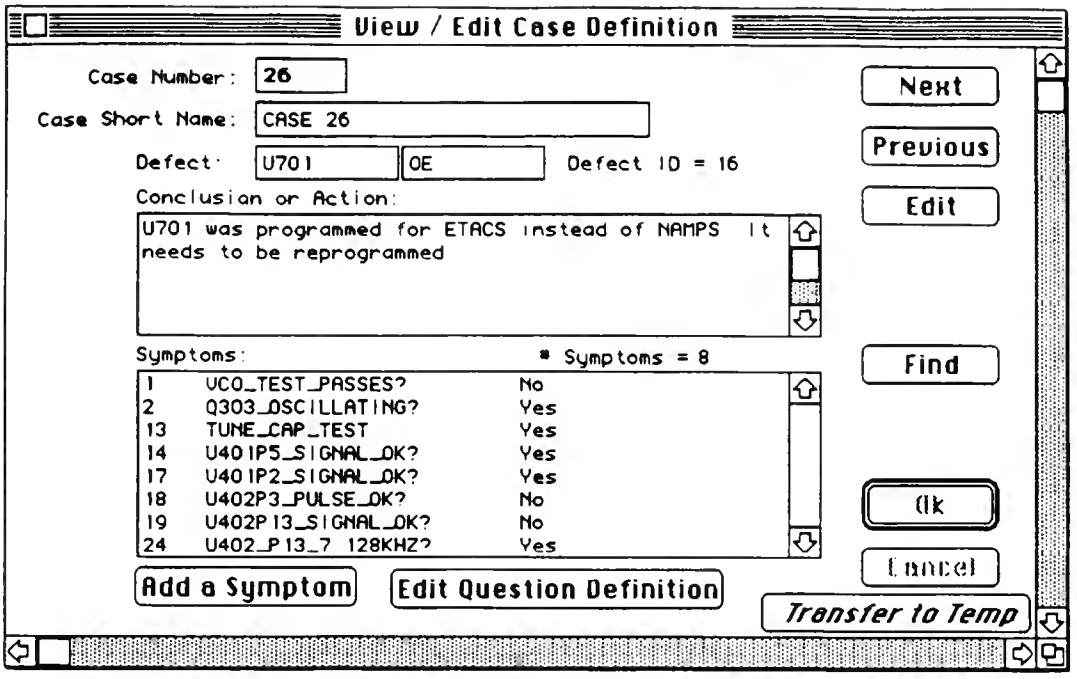


Figure 4c. Defining a new case with the CDSS prototype.

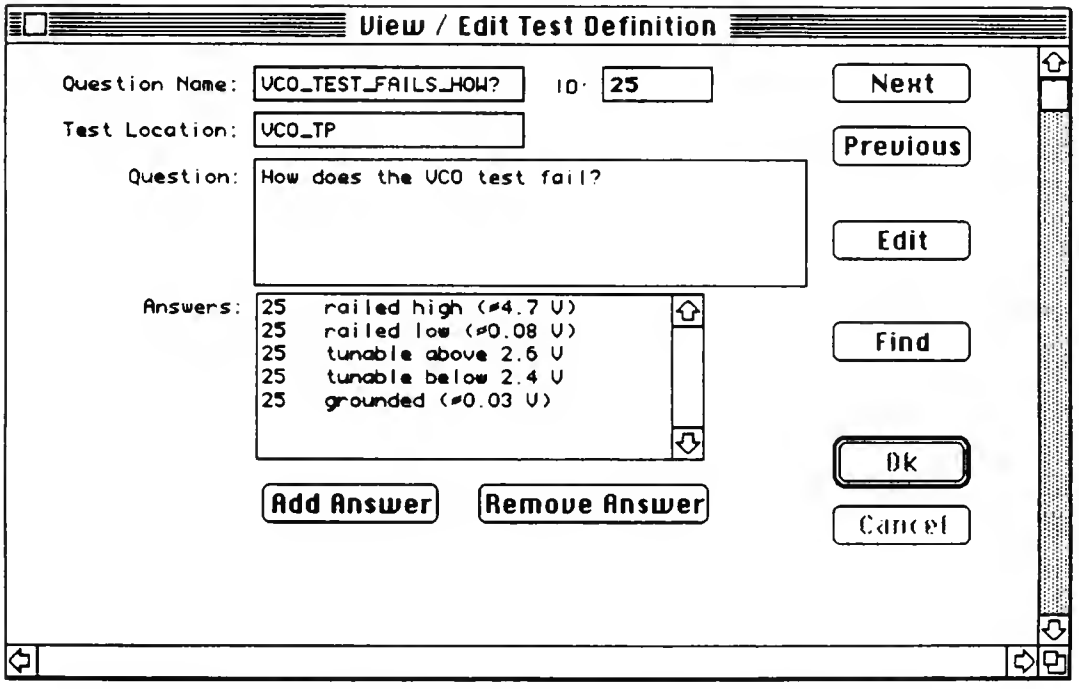
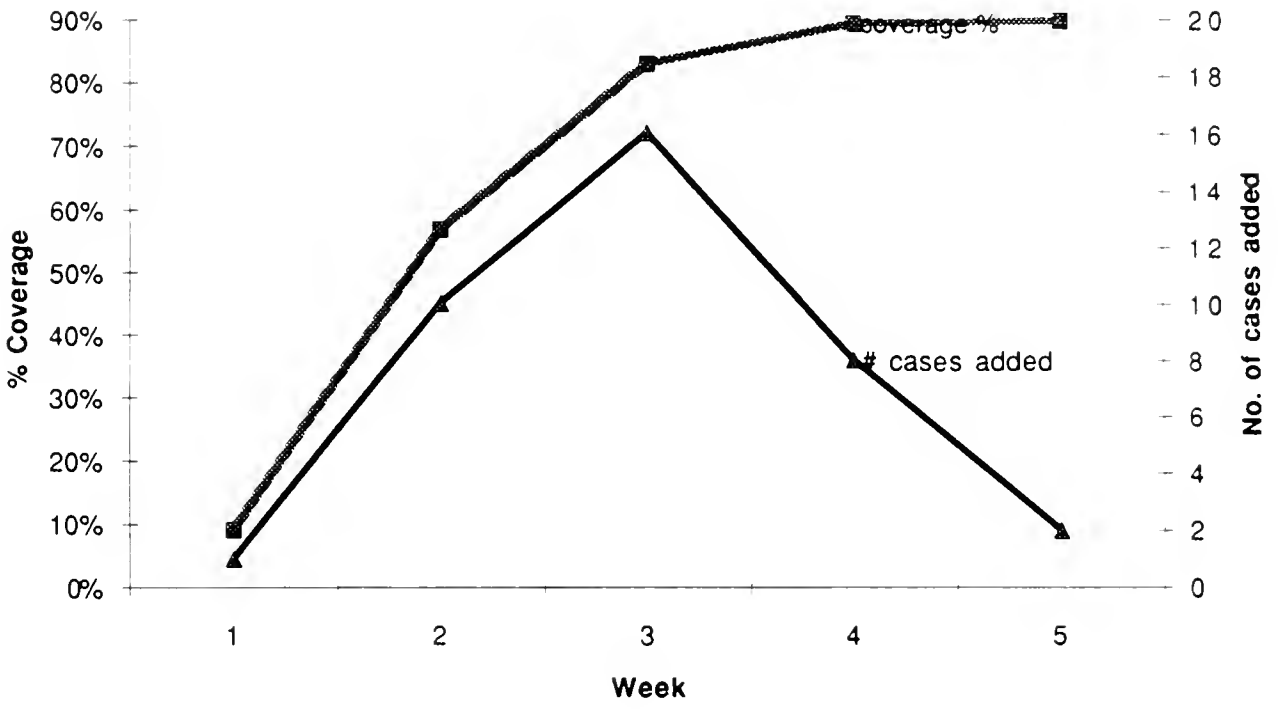


Figure 4d. Creating a new test definition using the CDSS prototype.

Figure 5: Cumulative coverage and number of cases added to CDSS



References

Bennetts, R.G., 1981. Introduction to Digital Board Testing. Crane, Russak & Co.

Butler, D. A., and G. J. Lieberman, 1984. "Inspection Policies for Fault Location", *Operations Research*, Vol. 32, No. 3, pp. 566-574.

Chevalier, P., 1992. "Optimal Inspection for Circuit Board Assembly", Part II of Ph.D. thesis, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Davis, R., 1984. "Diagnostic Reasoning Based on Structure and Behavior," Artificial Intelligence, Vol. 24, pp.347-410.

Davis, R., and W. C. Hamscher, 1988. "Model-Based Reasoning: Troubleshooting," A.I. Memo No. 1059, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

de Kleer, J., and B.C. Williams, 1987. "Diagnosing Multiple Faults," Artificial Intelligence, Vol. 32, pp.97-130.

Frank, W., and B. Sauve, 1989. "Expert Systems for Manufacturing and Testing Applications," Electrical Communication, Volume 63, Number 2.

Garcia-Rubio, M., 1988. "Key Problems and Solutions in Electronics Testing," Productivity and Quality Improvements in Electronics Assembly, J. A. Edosomwan and A. Ballakur (eds.), McGraw Hill Book Company.

Gensereth, M.R., 1984. "The Use of Design Descriptions in Automated Diagnosis," Artificial Intelligence, Vol. 24, pp. 411-436.

Hamscher, W.C., 1988. "Model-Based Troubleshooting of Digital Systems," A.I. Technical Report No. 1074, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Harmon, P., and D. King, 1985. Artificial Intelligence in Business: Expert Systems. John Wiley and Sons, Inc., New York.

Henneman, and Rouse, 1984. "Measures of human problem solving performance in fault diagnosis tasks," IEEE Transactions on Systems, Man, Cybernetics, SMC 14, pp. 99-112.

Ishida, Y., and H. Tokumaru, 1990. "A framework for case-based diagnosis — Dynamic Frame and Reasoning on the Representation," Transactions of the Society of Instrument and Control Engineers, Japan.

Kritz, J., and H. Sugaya, 1986. "Knowledge-Based Testing and Diagnosis of Analog Circuit Boards," FTCS Digest of Papers. 16th Annual International Symposium on Fault-Tolerant Computing Systems, pp. 378-383.

Lee, C.N., and P. Liu, 1988. "Case-based Reasoning for Robotic Assembly Cell Diagnosis," Proceedings of the SPIE — The International Society for Optical Engineering, SPIE Vol.1008, 241-246.

Longtin, M., L. M. Wein, and R. Welsch, 1992. "Sequential Screening in Semiconductor Manufacturing, II: Exploiting Spatial Dependence", Working paper, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Maunder, C.M., and R.E. Tulloss, 1990. The Test Access Port and Boundary-Scan Architecture, IEEE Computer Society Press Tutorial.

Moore, J. C., W. B. Richmond, and A. B. Whinston, 1988. "A Decision Theoretic Approach to File Search", *Computer Science in Economics and Management*, Vol. 1, pp. 3-19.

Moore, J. C., and A. B. Whinston, 1986. "A Model of Decision-making with Sequential Information-Acquisition", *Decision Support Systems*, Vol. 2, pp. 285-307.

Nachlas, J.A., S. R. Loney, and B. A. Binney, 1990. "Diagnostic-Strategy Selection for Series Systems", *IEEE Transactions on Reliability*, Vol. 39, No. 3, pp. 273-279.

Newstead, M.A., and R. Pettipher, 1986. "Knowledge Acquisition for Expert Systems," Electrical Communication, Volume 60, pp. 115-121.

Noble, P.J.W., 1989. Printed Circuit Board Assembly, Open University Press (Robotics Series)

Ou, J., and L. M. Wein, 1992. "Sequential Screening in Semiconductor Manufacturing, I: Exploiting Lot-to-lot Variability", Working paper, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Pappu, S., and S. C. Graves, 1992. "A Dual-ascent Algorithm for Determining the Optimal Testing Strategy", unpublished manuscript, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Preist, C., and B. Welham, 1990. "Modelling Bridge Faults for Diagnosis in Electronic Circuits," Working Notes of First International Workshop on Principles of Diagnosis (unpublished), Stanford University, Stanford, California.

Reiter, R., 1987. "A Theory of Diagnosis from First Principles," Artificial Intelligence, Vol. 32, pp. 57-95.

Rissland, E.L., and D.B. Skalak, 1988. "Case-based reasoning in a rule-governed domain," Proceedings - Fifth Conference on Artificial Intelligence Applications, IEEE.

Rouse, W.B., and N.M. Morris, 1985. "Review and Evaluation of Empirical Research in Troubleshooting," Journal of the Human Factors Society, Vol. 27, No. 5.

Rouse, W.B. and R.M. Hunt, 1986. "Human Problem Solving in Fault Diagnosis Tasks," Research Note 86-33, U.S.Army Research Institute for the Behavioral and Social Sciences.

Semmelbauer, T., 1992. "A Case-based Approach for Diagnosing Failed Printed Circuit Boards in Manufacturing", Masters' thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Soederberg, E. M., 1992. "Using Defect Data to Characterize Assembly Process Capabilities and Constraints for Printed Circuit Board Design for Assembly", Masters' thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Tong, D.W., E. Walther, and K.C. Zalondek, 1989. "Diagnosing an Analog Feedback System Using Model-Based Reasoning," Artificial Intelligent Systems in Government, Proceedings of the IEEE.

Trice, A., and R. Davis, 1989. "Consensus Knowledge Acquisition," A.I. Memo No. 1183, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Wenstrup, D. J., 1991. "Material Identification and Tracking in Manufacturing", Masters' thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Williams, T. W., and K. P. Parker, 1983. "Design for Testability–A Survey", *Proceedings of the IEEE*, Vol. 71, No. 1, pp. 98-112.

Yost, R.E., 1989. "Application of Model Based Reasoning to Diagnosis of Faults in Intertial Navigation Equipment," Master's Thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio.